

REMARKS

Reconsideration of this application, as amended, is respectfully requested.

Claims 1-21 are pending. Claims 1-21 stand rejected. Claims 1, 8, 15, and 16 have been amended. Claim 22 has been added. Support for the amendments is found in the specification, the drawings, and in the claims as originally filed. Applicants submit that the amendments do not add new matter.

Rejections Under 35 U.S.C. § 102(a) & 102(e)

Claims 1, 8 and 15 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,219,678 , of Yelland, et al. ("Yelland"). The Examiner stated that

Regarding claim 1, Yelland discloses all the claimed subject matter including accessing a reference array referencing at least one data object having a content stored in a corresponding memory location (see column 4, lines 23-32), determining a new memory location for the contents of each of the at least one data object and copying the contents of the at least one data object directly to the new memory location (see column 4, lines 35-46). Clearly in the process, the new data object contents of each new data object does not get stored to a cache.

(p. 3, Office Action 1/20/04)

Yelland discloses

An important concept in memory management is the manner in which memory is allocated to a task, deallocated, and then reclaimed. Memory deallocation and reclamation may be explicit and controlled by an executing program, or may be carried out by another special purpose program which locates and reclaims memory which is unused, but has not been explicitly deallocated. "Garbage collection" is the term used in technical literature and the relevant arts to refer to a class of algorithms utilized to carry out storage management, specifically automatic memory reclamation. There are many known garbage collection algorithms, including reference counting, mark-sweep, and generational garbage collection algorithms. These, and other garbage collection techniques, are described in detail in a book entitled "Garbage Collection, Algorithms For Automatic Dynamic Memory Management" by Richard Jones and Raphael Lins, John Wiley & Sons, 1996.

An object may be located by a "reference," or a small amount of information that can be used to access the object. One way to implement a reference is by means of a "pointer" or "machine address," which uses multiple bits of information, however, other

implementations are possible. General-purpose programming languages and other programmed systems often use references to locate and access objects. Such objects can themselves contain references to data, such as integers or floating-point numbers, and to yet other objects. In this manner, a chain of references can be created, each reference pointing to an object which, in turn, points to another object.

A subclass of garbage collectors known as "relocating" or "copying" garbage collectors, relocates objects that are still reachable by an executing program. Relocation of an object is accomplished by making a copy of the object to another region of memory, then replacing all reachable references to the original object with references to the new copy. The memory occupied by the original object may then be reclaimed and reused. Relocating garbage collectors have the desirable property that they compact the memory used by the executing program and thereby reduce memory fragmentation, which is typically caused by non-compacting garbage collectors.

(Col. 4, lines 6-46)

Applicants respectfully submit that claim 1 is not anticipated by Yelland under 35 U.S.C.

102§(e). Claim includes the following limitations:

A method comprising:

accessing a reference array, the reference array referencing at least one data object, each of the at least one data object having a contents stored in a corresponding memory location;

determining a new memory location for the contents of each of the at least one data object; and

copying the contents of the at least one data object directly to the new memory location thus creating a new data object for each of the at least one data object, each new data object having a new data object contents, such that upon copying the contents of the at least one data object to the new memory location, the contents of each new data object does not get stored to a cache memory.

(Amended Claim 1) (emphasis added)

Applicants have amended claim 1 to correct a typographical error.

Applicants respectfully submit that Yelland does not disclose a scheme wherein the new data object contents is not stored to the cache. Such storage is irrelevant to the inventive concept of Yelland, which is a novel method for maintaining an association for an object. Yelland initially store a new object without space for the association. Subsequently, when an association is recorded for the object, it is temporarily stored in a data structure. Then during a garbage collection operation, space within the object is allocated for the association and the association is then stored in the allocated space.

As cited by the Examiner, Yelland is simply summarizing a known garbage collection scheme. As such, Yelland has no occasion to fully describe a known drawback of such schemes. Yelland's omission in describing the storage of the new data object to the cache cannot be construed as a disclosure that no such storage takes place. Such was the known state of the art at the time of Yelland and at the time of filing the present application.

Applicants respectfully submit that, for this reason, claim 1 is not anticipated by Yelland. Applicants respectfully request the Examiner to withdraw the rejection of claim 1 as anticipated by Yelland.

Given that claims 2 – 7 depend, directly or indirectly, from claim 1, applicants respectfully submit that claims 2 – 7 are, likewise not anticipated by Yelland. Given that claims 8 and 15 are similarly rejected and that claims 9 – 14 and claims 16 – 21 depend, directly or indirectly, from claims 8 and 15, respectively, applicants respectfully submit that claims 8 – 21 are, likewise, not anticipated by Yelland.

Rejections Under 35 U.S.C. § 103(a)

Claims 2, 9 and 16 stand rejected under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 6,219,678 of Yelland, et al. ("Yelland") in view of applicants' admitted prior art (AAPA) at pages 1-3.

Applicants respectfully submit that the combination of Yelland and AAPA does not disclose the limitation that the contents of each new data object does not get stored to a cache memory.

Claims 3-7, 10-14 and 17-21 stand rejected under 35 U.S.C. § 103 as being unpatentable over U.S. Patent No. 6,219,678 of Yelland, et al. ("Yelland") in view of applicants' admitted prior art (AAPA) at pages 1-3, further in view of U.S. Patent No. 6,356,270 of Pentkovski, et al. ("Pentkovski") of record.

Applicants respectfully submit that the combination of Yelland, AAPA, and Pentkovski does not disclose the limitation that the contents of each new data object does not get stored to a cache memory.

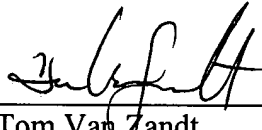
For the reason discussed above, in reference to Yelland, applicants respectfully submit that claims 1 –21 are not rendered obvious by any of the cited references alone or in combination.

It is respectfully submitted that in view of the amendments and arguments set forth herein, the applicable rejections and objections have been overcome. If there are any additional charges, please charge Deposit Account No. 02-2666 for any fee deficiency that may be due.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: 5/19/04

By: 
Tom Van Zandt
Reg. No. 43,219

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 720-8598

RECEIVED

MAY 25 2004

Technology Center 2100